文章编号:1671-8860(2005)12-1084-05　　**文献标志码**:A

基于空间网格和 Hilbert R-tree 的二级 R-tree 空间索引

郭　晶[1]　刘广军[2]　董绪荣[1]　郭　磊[3]

(1　北京总装备部指挥技术学院,北京市怀柔区京加路 1 号,101416)
(2　北京跟踪与通信技术研究所,北京市海淀区北清路 26 号,100094)
(3　石家庄铁道学院,石家庄市桥东区北二环路 17 号,050043)

摘　要:针对分布式海量空间数据库管理要求,提出了一种基于空间划分网格、Hilbert R-tree 和普通 R-tree 的二级空间索引结构,该结构被命名为 H2R-tree。然后,详细讨论了该结构的优点,并给出了实现算法。实际算例表明,H2R-tree 具有多方面的优良性能,是一种值得推广的二级索引技术。

关键词:空间索引;地理信息系统;R-tree;H2R-tree
中图法分类号:TP311; P208

空间数据的快速索引是实现海量数据管理的支撑技术之一,R-tree 索引[1]及其变种[2~4]是当前主流索引方法。R-tree 建立通常有两种途径:OBO(one by one)方法和批建立方法[4~6]。海量空间数据管理常常需要建立二级索引。在分布式数据服务环境下,一种可能的情况是,局部数据的 R-tree 索引已经存在并且正在运行,而更高层的应用需要对更大范围的数据进行管理。如图 1 所示,图 1 中的图区 A、B、C、D 共同组成了完整图区,而 A、B、C 实际上存放于局部数据服务器中。在这种情况下,需要建立一种高效的二级 R-tree 索引结构,既符合栅格空间管理习惯,又能够对局部的 R-tree 索引有效管理,不会造成存储冗余和牺牲查询效率。更重要的是,海量空间数据库经常处于多用户环境中,因此,数据的更新应该限制在局部区域,对主服务器的索引结构造成的影响应尽量小,并且更新迅速。

针对图 1 的分布式海量空间数据库的管理,本文将提出一种二级索引结构。首先定义空间数据的直接索引为第一级索引,而以第一级索引为目标再建立的索引称为二级索引。新结构主要满足以下要求。

1) 支持 R-tree 作为第一级索引,没有冗余存储,查询效率高。

2) 第二级索引建立过程符合数据网格划分习惯,允许建立者根据需要划分。具体到每个网格,可以根据数据分布,建立任意高度和节点容量的高质量第一级 R-tree。

3) 支持局部更新,局部更新造成的锁定、索引结构整理等影响被尽可能局限于局部网格。

4) 支持批建立和批插入。

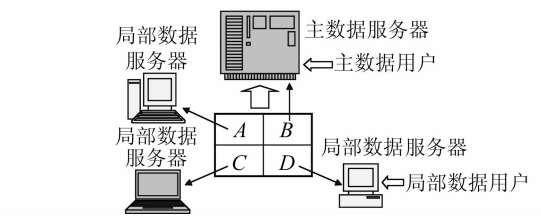

图 1　分布式空间数据服务

Fig. 1　Distributed Spatial Data Service

1　原理与算法

1.1　基本原理

新的二级索引结构称为 H2R-tree(Hilbert 2-Level R-tree),是一种基于 Hilbert R-tree 的二级索引技术,其基本思想是将空间数据按照空间网格进行划分,每个网格分别建立独立的 R-tree,形成第一级索引;而后将各网格按照中心 Hilbert 值进行分组,以网格内的第一级 R-tree 作为数据对象,采用 Hilbert R-tree 建立第二级索引。其

收稿日期:2005-11-20。

特点是叶节点中存放的不是空间数据指针,而是独立的没有等高度要求的第一级 R-tree 的指针。H2R-tree 的原理图如图 2 所示。

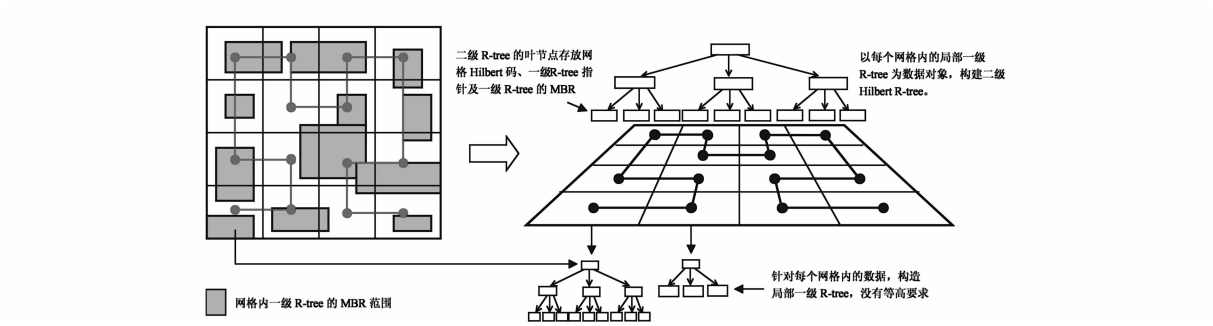

图 2　H2R-tree 原理示意图

Fig. 2　Structure of H2R-tree

1.2　Hilbert R-tree

空间数据沿着 Hilbert 曲线的线性编码称为 Hilbert 码。

基于 Hilbert 码的 R-tree 批建立思想是:先将待索引的空间数据按照 MBR 中心的 Hilbert 码值进行排序分组,然后按照自底向上的模式生成 R-tree。这种算法可以获得几乎近 100%的空间利用率,而且查询性能优于平方复杂度节点分裂的 R-tree、R^*-tree 和 Roussopoulos 的批建立方法[5]。

在此基础上,Kamel 提出了 Hilbert R-tree[6],从而新数据也可以利用 Hilbert 码在已建好的 R-tree 中寻找到合适的插入位置。Hilbert R-tree 与 R-tree 的区别在于节点,其具体描述为:Hilbert R-tree 的内部节点存放着若干形式为 Child-MBR、Child-Ref、Child-LHV 的数据项,其中 Child-MBR 和 Child-Ref 分别为子节点 MBR 和子节点指针,Child-LHV 则是子节点 MBR 包含的空间数据 MBR 中心的 Hilbert 码中的最大值。Hilbert R-tree 的叶节点则存放着若干形式为 Obj-MBR、Obj-Ref、Obj-HV 的数据项,其中 Obj-MBR 和 Obj-Ref 分别为空间数据 MBR 和空间数据指针,Obj-HV 则为空间数据 MBR 中心的 Hilbert 码。基于 Hilbert R-tree 的查询、插入与删除过程详见文献[4]。Hilbert R-tree 实现了在动态更新过程中仍然保持空间数据依照 Hilbert 码的排列顺序,而 H2R-tree 利用了这一特性,保证新数据的更新仅限于局部固定网格。

1.3　H2R-tree 结构与算法

1) 节点结构

第一级 R-tree 的数据结构与普通 R-tree 的节点数据结构相同。

第二级 Hilbert R-tree 内部节点结构与普通 Hilbert R-tree 的内部节点结构相同。

第二级 Hilbert R-tree 的叶节点与普通 Hilbert R-tree 叶节点略有差异,仅需存放 C_l 项(Cell-R-tree-Ref,Cell-HV)数据,其中 C_l 是叶节点容量,Cell-HV 是网格中心点 Hilbert 码,Cell-R-tree-Ref 是对第一级 R-tree 的引用,这种引用可以是文件指针,也可以是文件路径。特别地,如果某个网格内没有数据,则相应的一级 R-tree 建立为空树,空树的 MBR 设为无效 MBR,Cell-R-tree-Ref 指向该空树。值得注意的是,第二级 Hilbert R-tree 叶节点中没有存放数据项的 MBR,因为通过 Cell-R-tree-Ref 即可获得该 MBR,第二级 Hilbert R-tree 的叶节点的数据项中不再重复存放。

2) 建立

建立过程分为 4 个阶段:首先根据需要,由建立者划分空间规则网格,并求出各网格中心点 Hilbert 码。接着针对每个网格,由在 MBR 中心点中的空间数据建立第一级 R-tree 索引树,建立过程可采用任何一种 OBO 或 Bulk-Loading 方法;如果某个网格内没有数据,则建立空白第一级 R-tree,并设该第一级 R-tree 的根节点 MBR 为无效矩形,不参加父节点的 MBR 运算。第三阶段,将各第一级 R-tree 按照对应网格 Hilbert 值进行排列,分为 C_l 组,其中 C_l 为叶节点容量。最后,将各组第一级 R-tree 引用看作空间数据,生成第二级 Hilbert R-tree。

3) 查询

基于 H2R-tree 的查询与基于普通 R-tree 的查询类似,具体分为 4 个阶段:1) 第二级 Hilbert R-tree 的过滤查询,即通过查询窗口与节点 MBR 是否交叉重叠的检查,查到 Hilbert R-tree 的备选叶节点;2) 第二级 Hilbert R-tree 的精确查询,通过查

询窗口与叶节点内各项 Cell-R-tree-Ref 所指向的第一级 R-tree 的 MBR 进行是否交叉重叠的检查，查到与查询窗口有交叉重叠的各网格内的第一级 R-tree；3）第一级 R-tree 的过滤查询，通过进行查询窗口与节点 MBR 是否交叉重叠的检查，查到第一级 R-tree 的备选叶节点；4）第一级 R-tree 的精确查询，通过查询窗口与叶节点内各项 Obj-Ref 所指向的空间数据 MBR 进行是否交叉重叠的检查，查到与查询窗口有交叉重叠的各空间数据。

4）插入

H2R-tree 的插入分为以下两个阶段。

第一阶段：根据待插入数据的中心坐标值，将新数据分配到局部的网格中，其具体过程为：首先根据数据 MBR 的中心坐标计算数据的 Hilbert 码；然后利用数据 Hilbert 码，从第二级 Hibert R-tree 中查询到第一级 R-tree 作为待插入的目标。设其算法为 ChooseRT()，以类似 *C* 的伪代码描述如图 3 所示。

```
RT ChooseRT(int h)//RT 为返回的一级 R-tree，h 为为插入数据的 Hilbert 码
{
    N =ChooseLeaf(h);
    对于 N 中每个数据项(Cell-HV, Cell-R-tree-Ref)，计算|h-Cell-HV|;
    N =|h-Cell-HV|最小值对应的 Cell-R-tree-Ref;
    return N;
}
Leaf ChooseLeaf(int h) //Leaf 为返回的叶节点，h 为插入数据的 Hilbert 码
{
    N =Root_Node_Ref; //Root_Node_Ref 为根节点
    do
    {
      寻找大于 h 的最小 LHV 对应的子节点(Child-MBR, Child-Ref, Child-LHV);
      N =Child-Ref;
      if(N 是叶节点)
          return N;
      else //N 是非叶节点，则继续向下寻找合适的子节点
          continue;
}
```

图 3 查找第一级 R-tree 的算法

Fig. 3 Algorithm of Searching 1st R-tree

第二阶段：针对每个分配到新数据的网格，将新数据插入到对应的一级 R-tree 中。具体某个网格内的一级 R-tree 的批插入，可以采用 OBO、GBI、缓冲区树等任何一种动态方法。当然，如果局部网格内的数据量较小，也可以直接采用 Hilbert Bulk-Loading、STR 等一次性建立方法，将新老数据一起打破进行重建。

5）删除

基于 H2R-tree 的删除仍然是先执行查询过程，定位到存放该数据的第一级 R-tree，而后从该 R-tree 中删除，删除过程与普通 R-tree 的删除完全一样；如果删除导致该第一级 R-tree 的根节点 MBR 发生了改变，则必须更新存放该 R-tree 的第二级 Hilbert R-tree 的相应叶节点的 MBR，并向上更新所有受影响的 Hilbert R-tree 祖先节点的 MBR。但是，第一级 R-tree 的删除可能导致该第一级 R-tree 成为空树。对于这种情况，其处理途径为：继续保留该第一级 R-tree(物理上就是保留了一个空白索引文件)以及指向该 R-tree 的第二级 Hilbert R-tree 叶节点内的相应引用，只是将第一级空白 R-tree 的 MBR 置为无效 MBR；而后从第二级 Hilbert R-tree 的叶节点开始，更新各层祖先节点的 MBR，从而不会造成无效查询。如果多个 Hilbert 值相邻的网格内的第一级 R-tree 为空树，导致指向这些一级 R-tree 的二级 R-tree 的某个叶节点内所有数据项的 MBR 均为无效 MBR，则将该叶节点 MBR 置为无效 MBR，然后从父节点开始更新各层祖先节点的 MBR，直到根节点。

2 优势分析与实例

2.1 H2R-tree 的优势

1）符合网格管理惯例，却没有增加冗余。建立者可以根据需要划分网格，但每个数据只被索引一次。

2）与整株 R-tree 相比，H2R-tree 具有更高的查询效率。经典 R-tree 要求高度平衡，每个根到叶节点路径的层数一样，从而对于高度疏密不均的空间数据，无论查询窗口落到什么区域，查询过程都不得不游历相同层数，直到叶节点。而 H2R-tree 则可以克服此缺点：由于许多空间数据具有事先已知的地域分布特性，H2R-tree 的建立者可以根据经验将网格设计得更为合理。局部网格内的一级 R-tree 可以根据本网格内数据的个数建立不同高度的 R-tree，对于网格内没有数据的情况，甚至可以建立空白 R-tree。因此，如果查询窗口落于较稀的空间数据区，可以有效地减少查询游历高度，提高查询效率。

3）H2R-tree 实现了局部独立的高效率更新。无论插入还是删除，其影响被最大限度地限制到了网格内的第一级 R-tree：只有当第一级 R-tree 的根节点 MBR 发生改变，才会引起第二级 Hilbert R-tree 的相应叶节点的 MBR 更新；而且，由于采用了“空白 R-tree”和“无效 MBR”技

术,第二级 Hilbert R-tree 的结构不会发生改变。因此,由于 H2R-tree 局部更新造成的锁定主要在局部网格(图 4)。

4) H2R-tree 支持分布式数据管理、实现和操作简单。每个网格内的第一级 R-tree 以磁盘文件的方式存放于局部数据服务器,第二级 Hilbert R-tree 可以磁盘文件形式存放于主数据服务器,或直接建立为主存索引。H2R-tree 的查询操作与高度平衡的单级 R-tree 基本一致。

5) H2R-tree 支持批建立和批插入,兼容各种已有的批建立和批插入算法。

2.2　实例

图 5 为香港某地地形数据,共计 16 672 个线状地物数据,在图 5 数据范围内,数据分布密度严重不均。将图 5 所示数据按照 8×8 空间网格进行划分,示意为图 6。首先,由中心点落于某个网格内的全部空间数据构建局部第一级 R-tree,节点容量为 16,图 6 中各粗黑矩形框为各网格内第一级 R-tree 的 MBR。接着求出各网格中心点的 Hilbert 码,图 7 中绘制的曲线和浮点数字即为 Hilbert 曲线和 Hilbert 码。为适用于浮点坐标,采用了浮点 Hilbert 编码算法,其具体算法可参见文献[7]。图 8 中在网格内标注的数字分别为网格内数据个数和一级 R-tree 的高度。同时,利用图 4 中数据,还建立了一株完整的 Hilbert R-tree 用于测试比较,该 Hilbert R-tree 节点容量仍为 16,建立后树高 5 层。由图 8 可见,许多数据稀疏的网格内 R-tree 高度均小于 5,因此,落于这些区域中的查询效率将高于整株 Hilbert R-tree 的查询效率。

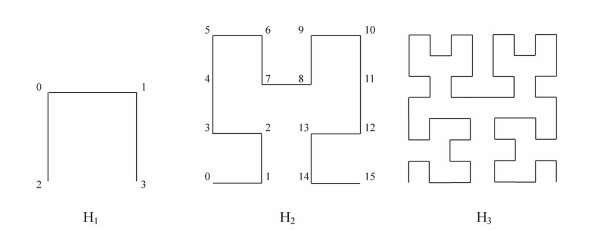

图 4　1,2 和 3 阶 Hilbert 曲线

Fig. 4　Hilbert Curves of Order 1,2 and 3

为验证该结论,测试了两类在图 5 范围内平均分布的开窗查询:第一类查询窗口矩形的高和宽分别小于网格的高和宽;第二类查询窗口的矩形的高和宽分别大于网格的高和宽。测试表明,基于 H2R-tree 的查询所需的磁盘操作次数小于基于整株 Hilbert R-tree 的磁盘操作次数,第一类查询非常明显,平均效率提高近 30%;第二类查询效率提高程度与查询窗口的高和宽大于网格高和宽的幅度反相关,但总是优于整株 Hilbert R-tree 的查询效率。

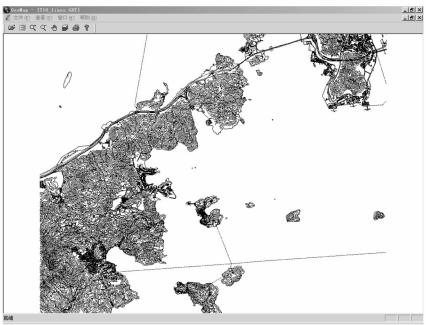

图 5　例图,数据总个数为 16 672

Fig. 5　Sample Map with 16 672 Data

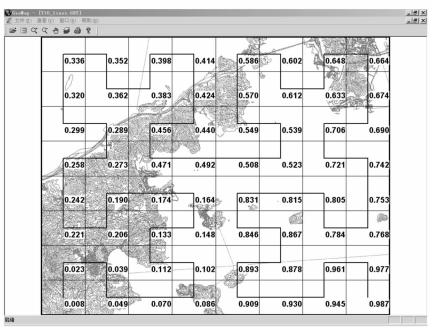

图 7　Hilbert 曲线及浮点 Hilbert 码

Fig. 7　Hilbert Curve and Float Hilbert Codes

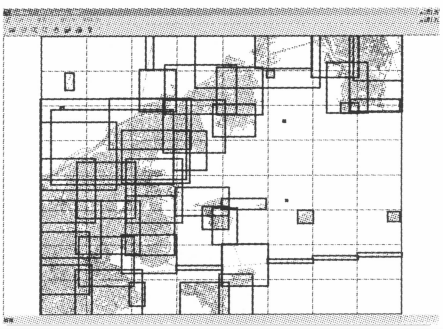

图 6　经网格分割后,各网格内数据的 MBR

Fig. 6　MBRs in the Cells After Grid Division

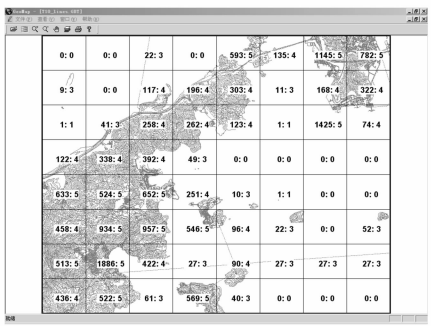

图 8　各网格内的数据个数和 R-tree 高度

Fig. 8　Data Count and R-tree Height in Each Cell

参 考 文 献

1 Guttman A. R-trees: a Dynamic Index Structure for Spatial Searching. ACM SIGMOD Conference, Boston, USA, 1984

2 Sellis T, Roussopoulos N, Faloutsos C. R^+-Tree: A Dynamic Index for Multi-Dimensional Objects. The 13th VLDB Conference, Athens, Greece, 1987

3 Beckmann N, Kriegel H P, Schneider R, et al. The R^*-tree: An Efficient and Robust Access Method for Points and Rectangles. ACM SIGMOD Conference, Atlantic City, USA, 1990

4 Kamel I, Faloutsos C. Hilbert R-tree-an Improved R-tree Using Fractals. The 20th VLDB Conference, Santiago, USA, 1994

5 Kamel I, Faloutsos C. On Packing R-trees. The 2nd CIKM Conference, Washington D C, USA, 1987

6 Leutenegger S, Edgington J M, Lopez M A. STR-a Simple and Efficient Algorithm for R-tree Packing. The 13th IEEE ICDE Conference, Birmingham, England, 1997

7 Roussopoulos N, Leifker D. Direct Spatial Search on Pictorial Databases Using Packed R-trees. ACM SIGMOD Conference, Austin, USA, 1985

8 郭 菁, 郭 薇, 胡志勇. 大型 GIS 空间数据库的有效索引结构 QR 树. 武汉大学学报·信息科学版, 2003, 28(3): 306～310

第一作者简介:郭晶,博士生。主要从事分布式移动 GIS 和 GPS 应用方向的研究。

E-mail:xixiguoo@sina.com

2-Level R-tree Spatial Index Based on Spatial Grids and Hilbert R-tree

GUO Jing[1] *LIU Guangjun*[2] *DONG Xurong*[1] *GUO Lei*[3]

(1 Beijing Institute of Command and Technology of Equipment, 1 Jingjia Road, Huairou Distrist, Beijing 101416, China)

(2 Beijing Institute of Tracking and Telecommunications Technology, 26 Beiqing Road, Haidian Distrist, Beijing 100094, China)

(3 Shijiazhuang Railway Institute, 17 Beierhuan Road, Qiaodong Distrist, Shijiazhuang 050043, China)

Abstract: Multi-level spatial index techniques are always used in the management of large spatial databases. This paper presents a novel 2-level index structure, which is based on the schemas of spatial grid-file, Hilbert R-tree and common R-tree. This new structure is named H2R-tree, and detailed algorithms are given. Using real data for test, the new method is proved to show superior performances in several aspects. The first, it suits for grid management with no additional demand; the second, H2R-tree shows better query efficiency; the third, it supports local independent update; the forth, it is suitable for distributed data management, and easy for realization; and the last, former bulk-loading methods can be applied in H2R-tree easily. Generally, H2R-tree is specifically suitable for the indexing of highly skewed, distributed, and large spatial database.

Key words: spatial index; GIS; R-tree; H2R-tree

About the first author: GUO Jing, Ph.D candidate. She is studying on the distributed mobile GIS and GPS applications.

E-mail: xixiguoo@sina.com